Library Management System

1. Installing Required Packages: -

* bcrypt
* cloudinary
* cookie-parser
* cors
* dotenv
* express
* express-fileupload
* mongoose
* node-corn
* nodemailer

1. Setting Up the Projects:

* Create app.js file
* Modify package. json
* Set express in app.js
* Create .env file where all the configuration will be used.

1. Database and Middleware Setup: -

* Create a database folder with a db.js file for databse connection and import it in app.js
* Create a middleware folder with errorMiddleware.js and import in app.js
* Add an error-handling (catchAsyncErrors) middleware to0 catch errors.

1. User Authentication System: -

* Create a model’s folder and add userModels.js.
* Create a controller’s folder and add authControllers.js for authentication functions.
* Implement a register function and add :

1. generateVerificationCode () in userModels.js
2. sendVerificationCode () in utils/sendVerificationCode.js

* Create generateCerificationOtpEmailTemplate(verificationCode) in utils/emailTemplates.js.
* Implement sendEmail() in utils/sendEmail.js and add following environment variables to your .env file.  
  ---SMTP\_HOST = smtp.gmail.com

--- SMTP\_SERVICE = gmail

---SMTP\_PORT = 465

---SMTP\_MAIL = your email id  
---SMTP\_PASSWORD

* Create an authRoutes.js file in routes and add a register route.
* Import and use this router in app.js
* Test the register route using Postman.

1. Authentication and Authorization: -

* Add verifyOTP in authController.js
* Create sendToken() in utils/sendToken.js and generateToken() in userModel.js and add the following environment variables to your .env file.

1. JWT\_SECRET\_KEY
2. JWT\_EXPIRE
3. COOKIE\_EXPIRE

* Test the verifyOTP route in Postman
* Implement login and logout routes and test them
* Create authMiddleware.js with isAuthenticated function.
* Apply this middleware to the logout function and test it.
* Add a getUser route and test it.